COMPUTABLY ENUMERABLE BOOLEAN ALGEBRAS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ying-Ying Tran

May 2018

COMPUTABLY ENUMERABLE BOOLEAN ALGEBRAS

Ying-Ying Tran, Ph.D.

Cornell University 2018

We study computably enumerable boolean algebras, focusing on Stone duality and universality phenomena. We show how classical Stone duality specializes to c.e. boolean algebras, giving a natural bijection between c.e. boolean algebras and Π^0_1 classes. We also give a new characterization of computably universal-homogeneous c.e. boolean algebras, which yields a more direct proof of the computable isomorphism between the Lindenbaum algebras of theories which satisfy the hypotheses of the second incompleteness theorem.

BIOGRAPHICAL SKETCH

Ying-Ying attended the Ross Mathematics Program from 2005 to 2009 and completed their B.S. in mathematics, despite proclaiming at the end of 2005 that they would not return to Ross and would definitely never major in math.

ACKNOWLEDGEMENTS

When looking back on history, we have a tendency to construct it as a chain of causes and effects which seem almost inevitable [3]. Getting this degree did not feel nearly so certain as I lived through it; instead it feels like a miracle of coincidences, the right words at the right times, and an unbelievably supportive network that I ended up here.

Two or three years into the program, I nearly quit. Thank you, Reyer Sjamaar, for being so sincerely surprised and quintessentially quizzical when I expressed my doubts. Thank you, Allen Knutson, for convincing me that if I left, I'd never return. Thank you, Richard Shore and Hadas Kress-Gazit, for your time and support. And thank you, Anil Nerode, for taking me on and declaring with utter confidence that I was a perfectly capable mathematician. There were several times where you asked the right questions at the right moment in the right way that shaped my direction and goals when I was otherwise faltering. Thank you for your boundless patience and energy. I was not an easy student, and I am ever grateful you stuck with me through it all.

Thank you, Ming-Ming, for being with me throughout despite the distance, steadfast and bright, for phone calls and vidchats and kitten pictures. Finally an enormous thank you to everyone I climbed with over the years, in and out of the department, in and out of Ithaca, be it once or many times. Thank you for meeting me where I was comfortable, for catching me (almost) every time I fell, for yelling encouragement up (or down) to me as I struggled with things (literally) beyond my reach, for cheering my triumphs and blaming the music when I failed, for making fun of me especially when I got too serious. For food, bad jokes, awkward conversations, laughs, cries, hugs, and surprises. Thank you for inspiring me to be a better climber and a better person.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In 1955 Myhill [8] showed every computably enumerable set is 1-reducible to every creative set. In the same paper he also showed that two sets are 1-reducible if and only if there is a computable permutation of $\mathbb{N}$ which maps one set into the other. Combining these with the second incompleteness theorem, Myhill proved that there is a computable "translation" between theorems of several systems, namely those to which the second incompleteness theorem apply, such as set theory and number theory.

We go in a different direction and show a computable isomorphism between the Lindenbaum algebras of set theory and arithmetic, by studying c.e. boolean algebras and computable universal-homogeneity. Pour-El and Kripke [9] proved this isomorphism as well. Montagna and Sorbi [7] generalized their proof to give a characterization of computably universal-homogeneous boolean algebras for the class of computably enumerable (c.e.) boolean algebras, namely effectively inseparable c.e. boolean algebras. However our proofs differ in notable ways. The different approach results in a different characterization of the computably universal-homogeneous c.e. boolean algebras, which we call Rosser-Turing boolean algebras. A possible advantage of our approach is that it avoids effective inseparability; the property of being Rosser-Turing follows more directly from arithmetization of Turing machines and the second incompleteness theorem. From computable universal-homogeneity, we can get an equivalence between effectively inseparable and Rosser-Turing c.e. boolean algebras.

We will also expand upon the classical Stone duality between boolean spaces

and boolean algebras. In an unpublished draft, Cenzer and Remmel [2] show that the prime ideals of a c.e. boolean algebra form a Π^0_1 class and that the clopen sets of a Π^0_1 class form a c.e. boolean algebra. We give more details of the duality between Π^0_1 classes and c.e. boolean algebras, including more detailed discussion of morphisms. Furthermore by viewing c.e. boolean algebras as quotients of a free algebra, we get a version of Stone duality with a structure-respecting bijection between c.e. boolean algebras and Π^0_1 classes. This bijection may be useful for exploring further connections between properties of c.e. boolean algebras and those of Π^0_1 classes.

1.1 Preliminaries

We briefly state some boolean algebra basics and their computable versions.

A *boolean algebra* $(B, \wedge, \vee, \neg, \equiv)$ is a nonempty set B with binary operations $\wedge$ and $\vee$, unary operation $\neg$, and an equivalence relation $\equiv$ on B satisfying the following identities:

1. associativity: $a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c, \quad a \vee (b \vee c) \equiv (a \vee b) \vee c$
2. commutativity: $a \wedge b \equiv b \wedge a, \quad a \vee b \equiv b \vee a$
3. distributivity: $a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c), \quad a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$
4. identities: $a \wedge 1 \equiv a, \quad a \vee 0 \equiv a$
5. annihilators: $a \wedge 0 \equiv 0, \quad a \vee 1 \equiv 1$
6. idempotence: $a \wedge a \equiv a, \quad a \vee a \equiv a$
7. absorption: $a \wedge (a \vee b) \equiv a, \quad a \vee (a \wedge b) \equiv a$
8. complementation: $a \wedge (\neg a) \equiv 0, \quad a \vee (\neg a) \equiv 1$

9. double negation: $\neg(\neg a) \equiv a$

10. DeMorgan's: $\neg a \wedge \neg b \equiv \neg(a \vee b), \quad \neg a \vee \neg b \equiv \neg(a \wedge b)$

This definition is strictly algebraic, in the style of Birkhoff algebras. We can define 0 and 1 not as constants but with identities $0 = a \wedge \neg a$ and $1 = a \vee \neg a$ for all $a \in B$. That is to say, we define it with equalities that hold for all elements in the domain, and in particular, we do not require that $0 \not\equiv 1$. This definition allows the *degenerate* boolean algebra in which $0 \equiv 1$. A *nondegenerate* boolean algebra is one in which $0 \not\equiv 1$.

A *filter* of a boolean algebra is a subset F such that for any $x, y \in F$ and $a \in \mathcal{B}$, $x \wedge y \in F$ and $a \vee x \in F$. An *ultrafilter* is a filter with the additional property that for any $a \in \mathcal{B}$, it is the case that $a \in F$ or $\neg a \in F$. An alternative definition of an ultrafilter is a boolean homomorphism from B to the 2-element boolean algebra.

In the context of Lindenbaum algebras, which will be of particular interest in Chapter 3, the boolean algebra equality is not known a priori but is a computably enumerable relation. Therefore, to define computably enumerable boolean algebras and computably enumerable boolean spaces we will think of $\equiv$ less as a partition of B into equivalence classes and more as a subset of $B \times B$. This way $\equiv$ can be a computable or computably enumerable subset of $B \times B$, for computable or computably enumerable boolean algebras respectively. It also avoids the complications of working with quotients of boolean algebras. Instead we can use a common domain for convenience and vary only the equivalence relation.

Definition 1.1. Let $\mathcal{B} = (B, \wedge, \vee, \neg, \equiv)$ be a boolean algebra. $(\mathcal{B}, \varphi)$ is a *com-*

putably enumerably represented (abbreviated c.e.-represented) boolean algebra if

1. φ is an injective map from B to $\mathbb{N}$ whose range is a computable subset of $\mathbb{N}$,
2. for any $p, q \in B$,
   (a) $\wedge$ and $\vee$ are functions from $B \times B$ to B such that the functions
   $$f_\wedge(\varphi(p), \varphi(q)) = \varphi(p \wedge q), \qquad f_\vee(\varphi(p), \varphi(q)) = \varphi(p \vee q)$$
   are computable,
   (b) $\neg$ is a function from B to B such that the function $f_\neg(\varphi(p)) = \varphi(\neg p)$ is computable, and
3. $\{(\varphi(p), \varphi(q)) : p \equiv q\}$ is a computably enumerable subset of $\varphi(B) \times \varphi(B)$.

$(\mathcal{B}, \varphi)$ is a *computably represented* boolean algebra if it is c.e.-represented and the equivalence is computable, i.e. $\{(\varphi(p), \varphi(q)) : p \equiv q\}$ is computable.

Though we require that a map from B to $\mathbb{N}$ exist (typically the Gödel numbering), if there is no risk of ambiguity we will suppress reference to the map. Throughout this work we will speak of "c.e. boolean algebras" and mean either a c.e.-represented boolean algebra or a boolean algebra where B is itself a computable subset of $\mathbb{N}$ whose presentation function φ will simply be the identity function. Note that every c.e.-represented boolean algebra is isomorphic to a boolean algebra with domain $\mathbb{N}$. In this spirit, sometimes we will talk as though functions are computable while acting directly on B, and sometimes we will talk as though B itself is (a subset of) $\mathbb{N}$.

To state it succinctly, in a computably enumerable boolean algebra, the domain is computable, the operations are computable, and the equivalence relation is computably enumerable. This definition generalizes the Lindenbaum algebra of statements of first order theories. Decidable theories have computable Lindenbaum algebras. On the other hand, the Lindenbaum algebra of statements of first order arithmetic, set theory, or any other computably enumerable but not decidable theory is a computably enumerable but not computable boolean algebra.

Definition 1.2. Suppose $(\mathcal{B}, \varphi)$ is a c.e.-represented boolean algebra with equivalence $\equiv$. $(B', \wedge', \vee', \neg', \equiv')$ is a *computably enumerable subalgebra* if

1. $\varphi(B')$ is a computable subset of $\varphi(B)$,
2. $\wedge', \vee', \neg'$ are the restrictions of $\wedge, \vee, \neg$ respectively to B', and
3. $\{(\varphi(p), \varphi(q)) : p, q \in B', p \equiv' q\}$ is a subset of $\{(\varphi(p), \varphi(q)) : p, q \in B, p \equiv q\}$.

In particular B' is *not* required to be closed under $\equiv$; for $c \in B'$, there may be a b in B such that $b \equiv c$ but b is not in B'.

Definition 1.3. Suppose $\mathcal{B} = (B, \wedge, \vee, \neg, \equiv)$ and $\mathcal{B}' = (B', \wedge, \vee, \neg, \equiv')$ are boolean algebras. A *boolean homomorphism* $f : \mathcal{B}' \to \mathcal{B}$ is a map such that for all a, b in B',

1. if $a \equiv' b$, then $f(a) \equiv f(b)$;
2. $f(a \wedge b) \equiv f(a) \wedge f(b)$;
3. $f(a \vee b) \equiv f(a) \vee f(b)$;
4. $f(\neg a) \equiv \neg f(a)$.

If $(\mathcal{B}, \varphi)$ and $(\mathcal{B}', \varphi')$ are c.e.-represented boolean algebras, then f is *computable* if $\varphi \circ f \circ (\varphi')^{-1}$ is a partial computable map from $\mathbb{N}$ to $\mathbb{N}$.

1.2 Enumerating c.e. boolean algebras

When enumerating the c.e. subsets of $\mathbb{N}$, we can refer to each by the algorithm that enumerates it. A similar idea can be applied to c.e. boolean algebras, but we need to be more careful.

Let P be a countably infinite set of propositional letters $\{p_1, p_2, \dots\}$, and let $S(P)$ be the set of all statements generated by P using $\wedge, \vee, \neg$. Define $\equiv_S$ so that $p \equiv_S q$ if and only if p and q have the same truth table. Finally let $\mathcal{S} = (S(P), \wedge, \vee, \neg, \equiv_S)$. Assign Gödel numbers to strings in this alphabet in the usual way. Then P and $S(P)$ are computable sets of strings and $\wedge, \vee, \neg$ are computable functions on strings. Define $\equiv_S$ so that $p \equiv_S q$ if and only if p and q have the same truth table. Then $\mathcal{S} = (S(P), \wedge, \vee, \neg, \equiv_S)$ is a computably presented boolean algebra. This is a free boolean algebra on the computable set of free generators P. We will show that every c.e. boolean algebra is a quotient of $\mathcal{S}$. To do so, we first discuss boolean homomorphisms.

Definition 1.4. Let $\mathcal{B} = (B, \wedge, \vee, \neg, \equiv)$ be any boolean algebra. A *$\mathcal{B}$-valued truth assignment* is a map $a : P \to \mathcal{B}$. A *$\mathcal{B}$-valued truth valuation* is a homomorphism $v : \mathcal{S} \to \mathcal{B}$ and v *extends* a if $v(p_i) \equiv a(p_i)$ for each $p_i \in P$.

Proposition 1.5. *Any $\mathcal{B}$-valued truth assignment $a : P \to \mathcal{B}$ has a unique extension to a $\mathcal{B}$-valued truth valuation $v : \mathcal{S} \to \mathcal{B}$.*

Corollary 1.6. *Any computable $\mathcal{B}$-valued truth assignment $a : P \to \mathcal{B}$ has a unique extension to a computable $\mathcal{B}$-valued truth valuation $v : S \to \mathcal{B}$.*

A potential difficulty of handling homomorphisms between c.e. presented boolean algebras is keeping the domains and equivalences consistent, ensuring

that maps are "well-behaved" and equivalences act as expected. We sidestep this by keeping the domain and presentation function constant. When we form quotients of boolean algebras, only the equivalence relation changes. This preserves computational information and makes it simpler to reason about.

Theorem 1.7. *Let $\mathcal{B} = (B, \wedge, \vee, \neg, \equiv)$ be a c.e. boolean algebra. Let $a : P \to \mathcal{B}$ be any computable map. Then a can be uniquely extended to a computable homomorphism $f : \mathcal{S} \to \mathcal{B}$ which induces a c.e. equivalence relation on $\mathcal{S}$. Conversely, every c.e. equivalence relation on $\mathcal{S}$ determines a computable surjective homomorphism into a c.e. boolean algebra.*

Proof. A computable map $a : P \to B$ is a B-valued truth assignment. By Corollary 1.6, a can be uniquely extended to a computable B-valued truth valuation $f : \mathcal{S} \to \mathcal{B}$, which is a homomorphism. The equivalence relation $\equiv_f$ induced by f on $\mathcal{S}$ is defined by $p \equiv_f q$ if and only if $f(p) \equiv f(q)$. Since f is computable and $\equiv$ is c.e., we have $\equiv_f$ is also c.e.

Now suppose $\equiv$ is a c.e. equivalence relation on S and $\mathcal{B} = (S(P), \wedge, \vee, \neg, \equiv)$. Let $a : P \to \mathcal{B}$ send p_i to p_i. This is a $\mathcal{B}$-valued truth assignment, so can be uniquely extended to a $\mathcal{B}$-valued truth valuation v. Since the image of a is all of P and P generates B, v is surjective.

Uniqueness follows from the first isomorphism theorem. □

Theorem 1.7 allows us to present any c.e. boolean algebra as a computable quotient of $\mathcal{S}$.

Proposition 1.8. *From a c.e. boolean algebra $\mathcal{B} = (B, \wedge, \vee, \neg, \equiv)$, we can compute a surjective homomorphism from $\mathcal{S}$ onto $\mathcal{B}$ and a c.e. equivalence relation on $\mathcal{S}$.*

Proof. Because B is computably enumerable, we can enumerate the elements as $b_1, b_2, \ldots$, and assign $a(p_i) = b_i$ for each p_i in P. Thus we have a computable surjective $\mathcal{B}$-valued truth assignment a, which can be extended to a computable surjective homomorphism $f : \mathcal{S} \to \mathcal{B}$. As in Theorem 1.7, let $\equiv_f$ be defined by $p \equiv_f q$ if and only if $f(p) \equiv f(q)$. Since f is computable and $\equiv$ is c.e., $\equiv_f$ is also computably enumerable. □

We are now ready to enumerate the computably enumerable boolean algebras. First assign Gödel numbers to all pairs of elements of $S(P)$ in the usual way. Dovetail an algorithm for a standard computable enumeration of all c.e. subsets w_e of $S(P) \times S(P)$ with an algorithm for extending each w_e to its generated equivalence relation $\equiv_e$ on $S(P)$ to get a standard enumeration of all c.e. equivalence relations. This gives a standard enumeration of all c.e. boolean algebras (up to computable isomorphism) $\mathcal{S}_e = (S(P), \wedge, \vee, \neg, \equiv_e)$.

We emphasize once again that we never form equivalence classes. All $\mathcal{S}_e = (S, \wedge, \vee, \neg, \equiv_e)$ have the same domain $S(P)$ and the same boolean operations as $\mathcal{S}$. They differ only in the equivalence relation $\equiv_e$.

CHAPTER 2

STONE DUALITY FOR C.E. BOOLEAN ALGEBRAS

We will show a duality between c.e. boolean algebras and Π^0_1 classes. Then in section 2.5 we will modify the duality to show a bijection. Our view of this will be generally algebraic in flavor.

Since the Cantor set is $\{0,1\}^{\mathbb{N}}$, Π^0_1 classes are also Π^0_1 subsets of the Cantor set. We will use these two ideas of Π^0_1 classes interchangeably, primarily working with the Cantor set definition.

2.1 Definitions

We have already defined the relevant notions in the category of boolean algebras so here is the topological side.

Definition 2.1. A *boolean space* (also known as a *Stone space*) is a totally disconnected compact Hausdorff space.

In a boolean space, the clopen sets form a basis of the topology. Then the clopen sets form a boolean algebra (of sets).

The classical Stone duality is between boolean algebras and boolean spaces. For a boolean algebra $\mathcal{B}$, the dual boolean space $\mathcal{B}^*$ consists of points which are ultrafilters of $\mathcal{B}$. For a given element b of $\mathcal{B}$, let $b^* = \{s \in \mathcal{B}^* : s(b) = 1\}$. The topology on $\mathcal{B}^*$ is generated by clopen sets of the form b^* for $b \in \mathcal{B}$, and the isomorphism sends b to b^*.

There are many good write-ups of the interesting parts of the duality (for example [4]), so we will only note one proposition that will be useful later.

Proposition 2.2. *Let $\mathcal{B}$ be a boolean algebra with Stone dual $\mathcal{B}^*$ and a clopen basis which consists of sets of the form $b^* = \{s \in \mathcal{B}^* : s(b) = 1\}$ for $b \in \mathcal{B}$. Then every clopen set is a basic clopen set.*

Proof. Since every clopen set is a finite union of sets in the clopen basis, it suffices to show that $b_1^* \cup b_2^* = (b_1 \vee b_2)^*$. The left is a subset of the right: if $s(b_i) = 1$, then $s(b_1 \vee b_2) = s(b_1) \vee s(b_2) = 1$. To show the right is a subset of the left, suppose $s(b_1 \vee b_2) = 1$. If $s(b_1) = s(b_2) = 0$, then $s(b_1 \vee b_2) = s(b_1) \vee s(b_2) = 0$. Therefore $s(b_1) = 1$ or $s(b_2) = 1$. □

Each point of $\mathcal{B}^*$, i.e. ultrafilter of $\mathcal{B}$, is the intersection of all clopen sets which contain it, so in some sense can be viewed as the limit of a sequence of clopen sets. Hence any computational information about the point is in the computational content of the sequence of clopen sets. We define a computably enumerable boolean space and computably continuous map accordingly.

Definition 2.3. A *computably enumerably represented* boolean space T is a boolean space with a computable enumeration φ of the clopen sets and a c.e. equivalence relation $\equiv$ on the dual boolean algebra T^* such that (T^*, φ) is a c.e.-represented boolean algebra.

Definition 2.4. A computably continuous map between c.e. boolean spaces is a continuous map for which there is an algorithm which gives the preimage of a basic clopen set as the finite union of basic clopen sets.

Though we primarily use the dual in its classical form, we briefly define Π_1^0 classes and explain the relation to the Cantor set.

Definition 2.5. A formula (in the language of second-order arithmetic) is Π^0_1 if it is logically equivalent to a formula of the form $\forall n_1 \forall n_2 \cdots \forall n_k \psi(X, n_1, n_2, \ldots, n_k)$, where ψ is an arithmetical formula with only bounded quantifiers, i.e. quantifiers of the form $\forall n < t$ or $\exists n < t$ and X is a free set variable.

Note that bounded quantifiers add no computational complexity, since we assume $\mathbb{N}$ to be the domain; if ψ is computable, then so are $(\forall n < t)\psi$ and $(\exists n < t)\psi$. Then if the innermost formula ψ is computable, a Π^0_1 formula is co-c.e. We check each $n \in \mathbb{N}$, and if $\psi(n)$, then we keep going, but if $\neg\psi(n)$, then the formula is refutable.

Definition 2.6. A class A, i.e. a set of subsets of $\mathbb{N}$, is *defined by a formula* φ if the elements of A are exactly those sets which satisfy φ, i.e. $X \in A$ if and only if $\mathbb{N}, 2^{\mathbb{N}} \vDash \varphi(X)$. A is a Π^0_1 class if it is defined by a Π^0_1 formula.

A class defined by a formula with only bounded quantifiers is computable: for each $t \in \mathbb{N}$, we simply check each $n < t$. Since there are only finitely many to check, we can determine in finite time whether or not X is in the class. However if the formula is Π^0_1, then we can determine if X is not in the set but not if t is in the set, since verifying that t is in the set requires checking every $n \in \mathbb{N}$. Hence Π^0_1 classes correspond exactly to co-c.e. classes.

Each subset of $\mathbb{N}$ corresponds to a two-valued map on $\mathbb{N}$, or a countably infinite string of 0's and 1's, so as mentioned at the start of the chapter, we can also think of a class as a subset of the Cantor set. The characterization of a subset of $\mathbb{N}$ as a countably infinite string of 0's and 1's suggests an infinite path through a binary tree, a representation which will appear as well. The Π^0_1 subsets of the Cantor set are also exactly the effectively closed subsets of the Cantor space [1].

Though a little more removed from the other representations, since one side of the duality is the category of topological spaces, this topological view of Π^0_1 classes will fit in naturally.

2.2 Computably enumerable boolean algebras to Π^0_1 classes

The computability of a c.e. boolean algebra is reflected by the computability of the formula which defines a class. Otherwise it goes the same way as the classical duality.

Theorem 2.7. *Classical Stone duality maps c.e. boolean algebras to Π^0_1 classes.*

Proof. Let $\mathcal{B} = (B, \wedge, \vee, \neg, \equiv)$ be a c.e. boolean algebra. A subset S of B is an ultrafilter if and only if it satisfies the sentence

$$(\forall x \forall y)(\varphi_1(x,y) \wedge \varphi_2(x,y) \wedge \varphi_3(x) \wedge \varphi_4),$$

where

$$\begin{aligned}
\varphi_1(x,y) &= (x \in S \wedge (x \wedge y \equiv x)) \Rightarrow y \in S) \\
\varphi_2(x,y) &= ((x \in S \wedge y \in S) \Rightarrow (x \wedge y) \in S) \\
\varphi_3(x) &= (x \in S \vee (\neg x) \in S) \\
\varphi_4 &= (0 \notin S \wedge 1 \in S).
\end{aligned}$$

Since $\mathcal{B}$ is c.e., $\wedge$, $\vee$, and $\neg$ are computable while $\equiv$ is c.e. S is given via oracle, so each φ_i is computable. Hence $\mathcal{B}^*$ is a Π^0_1 class. □

The computatibility of a homomorphism translates to a witness to the computability of continuity.

Theorem 2.8. *Under classical Stone duality, in the category of c.e. boolean algebras, computable boolean algebra homomorphisms are sent to computably continuous maps.*

Proof. Let F be the functor from the category of c.e. boolean algebras (where morphisms are computable boolean homomorphisms) to the category of Π^0_1 classes (where morphisms are computable continuous functions) which sends $\mathcal{B}$ to $\mathcal{B}^*$. We will now define how F sends morphisms. Suppose $f : \mathcal{B}_1 \to \mathcal{B}_2$ is a morphism. Let $F(f) : F(B_2) \to F(\mathcal{B}_1)$ send an ultrafilter t of $\mathcal{B}_2$ to the preimage of $t^{-1}(1)$ under f, i.e. $F(f)(t) = f^{-1}(t^{-1}(1))$. $F(f)$ is continuous by classical Stone duality.

The topology of $F(B_i)$ is generated by $b^* = \{s : s(b) = 1\}$ for $b \in B_i$ where s is an ultrafilter of $\mathcal{B}_i$. Therefore to show $F(f)$ is computably continuous, it suffices to show that the preimage of a basic clopen set is a basic clopen set. Let b^* be a basic clopen set of $F(\mathcal{B}_1)$, and suppose c^* is a basic clopen set of $F(\mathcal{B}_2)$ such that $F(f)(c^*) \subseteq b^*$. Note that $F(t)(c^*) = \{f^{-1}(t) : t(c) = 1\}$. Then b is in $f^{-1}(t)$ for every ultrafilter t sending c to 1. Equivalently $f(b)$ is in every ultrafilter which sends c to 1. However the intersection of all ultrafilters containing c is the singleton set $\{c\}$, so $c = f(b)$, so the preimage of the basic clopen set b^* is the basic clopen set $(f(b))^*$. Therefore $F(f)$ is computably continuous. □

2.3 Π^0_1 classes to computably enumerable boolean algebras

Before we define the functor from Π^0_1 classes to c.e. boolean algebras, recall that an alternate view of the Cantor set is as the set of infinite paths through an infinite binary tree. As shown in [1], every Π^0_1 subset K of the Cantor set is the

set of infinite paths through some computable tree T which is a subtree of the infinite binary tree. For a finite string σ, let $|\sigma|$ denote the length of σ and let $\sigma(n)$ be the $(n-1)$th element of σ. For finite strings σ and τ, we say σ is an initial segment of τ, denoted $\sigma \prec \tau$, if $\sigma(i) = \tau(i)$ for $0 \leq i < |\sigma|$. Further let $I(\sigma)$ be the set of all infinite strings with σ as an initial segment.

Definition 2.9. Suppose K is an effectively closed subset of the Cantor space $\mathbf{C}$. Let $RC(K)$ be the set of relatively clopen subsets of K. Define $\wedge$ to be union, $\vee$ to be intersection, and $\neg$ to be set-complement. $RC(K)$ is closed under these operations, and the operations are computable, so $\mathcal{RC}(K) = (RC(K), \wedge, \vee, \neg, =)$ is a boolean algebra with equality being set equality. The clopen subsets of $\mathbf{C}$ form a boolean algebra with the same operations, so let $\mathcal{B}(K)$ be the quotient algebra with $U =_K V$ if and only if $U \cap K = V \cap K$, for U, V clopen subsets of $\mathbf{C}$.

Lemma 2.10. *For a Π^0_1 subset K of $\mathbf{C}$, $\mathcal{B}(K)$ is a c.e. boolean algebra.*

Proof. Begin with a Π^0_1 subset K of the Cantor set, where K is the set of infinite paths through a computable tree T. Clopen subsets of $\mathbf{C}$ can be represented as finite unions of intervals, so if U, V are clopen subsets of $\mathbf{C}$, let

$$U = \bigcup_{i=1}^{m} I(\sigma_i), \qquad V = \bigcup_{j=1}^{n} I(\tau_j).$$

Then $U \cap K \subseteq V \cap K$ if and only if for every $i \leq m$, $I(\sigma_i) \cap K \subseteq V \cap K$. For any finite string σ, $I(\sigma) \cap K \subseteq V \cap K$ if and only if

$$(\exists n \geq |\sigma|)(\forall \tau)((|\tau| = n \;\&\; \tau \in T \;\&\; \sigma \prec \tau) \Rightarrow (\exists j)(\tau_j \prec \tau)).$$

Since T is computable, this expression is c.e., so $U \cap K \subseteq V \cap K$ is c.e. An identical argument shows that $U \cap K \subseteq V \cap K$ is also c.e., which completes the proof that $\mathcal{B}(K)$ is a c.e. boolean algebra. $\square$

Lemma 2.11. *For a Π^0_1 subset K of $\mathbf{C}$, $\mathcal{RC}(K)$ is computably isomorphic to $\mathcal{B}(K)$.*

Proof. Let $\varphi : \mathcal{B}(K) \to \mathcal{RC}(K)$ send $U/{=_K}$ to $U \cap K$. By definition of $=_K$, φ is well-defined, computable, and injective, so it remains to show surjectivity. Let A be a clopen subset of K. Then $A = U \cap K$ for some U open in $\mathbf{C}$. Since U is open in $\mathbf{C}$, we can express U as $\bigcup_{i \in I} I(\sigma_i)$, where I may not be finite. Since A is closed in K and K is closed in $\mathbf{C}$, A is compact. The union $\bigcup_{i \in I} I(\sigma_i)$ covers A, so there is a finite subset J of I such that $A \subseteq \bigcup_{i \in J} I(\sigma_i)$. Let $V = \bigcup_{i \in J} I(\sigma_i)$. As established previously, $A \subseteq V$ so $A \subseteq V \cap K$. On the other hand, $V \subseteq U$, so $A = U \cap K \supseteq V \cap K$. Thus V is a clopen subset of $\mathbf{C}$ such that $V \cap K = A$, proving that φ is surjective. □

Theorem 2.12. *Classical Stone duality maps Π^0_1 classes to c.e. boolean algebras.*

Proof. This follows directly from lemmas 2.10 and 2.11. Under classical Stone duality, if K is a Π^0_1 class, K is dual to $\mathcal{RC}(K)$ which is computably isomorphic to $\mathcal{B}(K)$. Since $\mathcal{B}(K)$ is a c.e. boolean algebra, so is the dual of K. □

Theorem 2.13. *Under classical Stone duality, in the category of c.e. boolean spaces, computably continuous maps are sent to computable boolean algebra homomorphisms.*

Proof. Let G be the functor from Π^0_1 classes to c.e. boolean algebras which sends K to $\mathcal{RC}(K)$ (as defined in the proof of Theorem 2.12). Suppose $g : K_1 \to K_2$ is a morphism. Let $G(g) : G(K_2) \to G(K_1)$ send a clopen subset c of K_2 to its preimage under g, i.e. $G(g)(c) = g^{-1}(c)$. $G(g)$ is a boolean homomorphism by classical Stone duality. A clopen subset c of K_2 is given in terms of the basic clopen sets, which generate $G(K_2)$. Since g is computably continuous, $G(g)(c) = g^{-1}(c)$ is a finite union of basic clopen sets of K_1, i.e. generating elements of $G(K_1)$. Therefore $G(g)$ is a computable homomorphism. □

2.4 More on morphisms

We will show that computability of maps is preserved by Stone duality. In doing so, we will reproduce parts of the classical proofs. First we prove a lemma relating the kernel of a boolean homomorphism and the image of its dual continuous map.

Lemma 2.14. *Let $f : \mathcal{B} \to \mathcal{A}$ be a boolean homomorphism with Stone dual $f^* : \mathcal{A}^* \to \mathcal{B}^*$. Then $b \in \ker f$ if and only if $(f^*)^{-1}(\{s \in \mathcal{B}^* : b \in s\}) = \varnothing$.*

Proof. First note that

$$\begin{aligned}(f^*)^{-1}(\{s : b \in s\}) &= \{t \in \mathcal{A}^* : f^*(t) \in \{s \in \mathcal{B}^* : b \in s\}\} \\ &= \left\{t : f^{-1}(t) \in \{s : b \in s\}\right\} = \left\{t : b \in f^{-1}(t)\right\} \\ &= \{t : f(b) \in t\}\,.\end{aligned}$$

Suppose $(f^*)^{-1}(\{s : b \in s\}) = \varnothing$. Then there are no ultrafilters containing $f(b)$. Every nonzero element a of $\mathcal{A}$ is in some ultrafilter, for instance $\{t \in \mathcal{A}^* : a \in t\}$, so $f(b) = 0$. On the other hand, if $b \in \ker f$, then $f(b) = 0$ and no ultrafilter contains 0, so $(f^*)^{-1}(\{s : b \in s\}) = \varnothing$. □

With this characterization of the kernel of a boolean homomorphism, we are ready to show that injective maps in one category correspond to surjective maps in the other category.

Theorem 2.15. *Under Stone duality,*

1. *injective (surjective) computable homomorphisms correspond to surjective (injective) computably continuous maps, and conversely;*

2. *computable isomorphisms correspond to computable homeomorphisms and conversely;*

3. *composition of computable homomorphisms corresponds to composition of the dual computably continuous maps in the opposite order.*

Proof. 1. It suffices to show this for injective and surjective homomorphisms. Computability follows from Theorems 2.8 and 2.13, so much of what follows is from the classical proof, parts of which are done in [4] but we do it more explicitly here.

We will first show the correspondence from boolean homomorphisms to continuous maps. Let $f : \mathcal{B}_1 \to \mathcal{B}_2$ be a boolean homomorphism, with dual $g : K_2 \to K_1$. Recall from Theorem 2.8 that for $b \in \mathcal{B}_1$ and $t \in K_2$, $f(b) \in t$ if and only if $b \in g(t)$.

First suppose f is injective. Then whenever P is clopen and $(f^*)^{-1}(P) = \varnothing$, we have $P = \varnothing$. This follows from Proposition 2.2 and Lemma 2.14. Then if P is clopen in $\mathcal{B}^*$ and P is a subset of $\mathcal{B}^* - \operatorname{im} f^*$, P has empty intersection with $\operatorname{im} f^*$, so $(f^*)^{-1}(P) = \varnothing$. Therefore $P = \varnothing$, so $\mathcal{B}^* - \operatorname{im} f^*$ contains no clopen subsets of $\mathcal{B}^*$ besides $\varnothing$. However $\mathcal{A}^*$ is compact so $f^*(\mathcal{A}^*)$ is compact and hence effectively closed, so $\mathcal{B}^* - \operatorname{im} f^*$ is open and a union of its clopen subsets. Since $\mathcal{B}^* - \operatorname{im} f^*$ contains no clopen subsets besides $\varnothing$, $\mathcal{B}^* - \operatorname{im} f^* = \varnothing$, which shows f^* is surjective (finally).

Suppose f is surjective and $g(s) = g(t)$ for some $s, t \in K_2$. Then for every $b \in \mathcal{B}_1$, we have $f(b) \in s$ if and only if $b \in g(s) = g(t)$ if and only if $f(b) \in t$. Thus $s \cap \operatorname{im} f = t \cap \operatorname{im} f$, but $\operatorname{im} f = \mathcal{B}_2$, so $s = t$ and g is injective.

Now we will show the correspondence from continuous maps to boolean homomorphisms. Let $g : K_1 \to K_2$ be a continuous map, with dual $f :$

$\mathcal{B}_2 \to \mathcal{B}_1$. Recall from Theorem 2.13 that for $s \in K_1$ and $c \in \mathcal{B}_2$, $g(s) \in c$ if and only if $s \in f(c)$.

First suppose g is injective and let b be an element of B_1, i.e. a clopen subset of K_1. Note g is a homeomorphism between K_1 and $\operatorname{im} g$, so $g(b)$ is clopen in $\operatorname{im} g$. Since $g(b)$ is open in $\operatorname{im} g$, there is an open subset S of K_2 such that $g(b) = S \cap \operatorname{im} g$. K_2 has a clopen basis $\{c_i : i \in I\}$, so there is some subset I' of I such that $S = \bigcup_{i \in I'} c_i$. Because K_1 is compact and g is continuous, $\operatorname{im} g$ is compact. Since $g(b)$ is effectively closed in $\operatorname{im} g$ and $\operatorname{im} g$ is compact, $g(b)$ is compact. Therefore there is a finite subset J of I' such that $g(b) \subseteq \bigcup_{i \in J} c_i$. Let $c = \bigcup_{i \in J} c_i$. Then $g(b) \subseteq c \cap \operatorname{im} g$, and $g(b) = S \cap \operatorname{im} g \supseteq c \cap \operatorname{im} g$, so $g(b) = c \cap \operatorname{im} g$. Furthermore c is clopen in K_2, so $c \in \mathcal{B}_2$. Therefore $f(c) = g^{-1}(c) = b$, and f is surjective.

Suppose g is surjective and $f(b) = f(c)$ for some $b, c \in \mathcal{B}_2$. Then for every $s \in K_1$, we have $g(s) \in b$ if and only if $s \in f(b) = f(c)$ if and only if $g(s) \in c$. Thus $b \cap \operatorname{im} g = c \cap \operatorname{im} g$, but $\operatorname{im} g = K_2$, so $b = c$ and f is injective.

2. This follows immediately from the previous part.

3. Suppose $f_1 : \mathcal{B}_1 \to \mathcal{B}_2$ and $f_2 : \mathcal{B}_2 \to \mathcal{B}_3$ are boolean homomorphisms with duals $g_1 : K_2 \to K_1$ and $g_2 : K_3 \to K_2$ respectively. The composition $f_2 f_1$ is a boolean homomorphism from $\mathcal{B}_1$ to $\mathcal{B}_3$, so its dual g is a continuous map from K_3 to K_1. For an element u of K_3, we have

$$g(u) = (f_2 f_1)^{-1}(u) = (f_1^{-1} f_2^{-1})(u) = (g_1 g_2)(u),$$

so the dual of composition is composition in the opposite order.

Now add the assumption that f_1 and f_2 are computable. Then $f_2 f_1$ is computable, so its dual is also computable, finishing the proof.

□

2.5 A natural bijection between c.e. boolean algebras and Π^0_1 classes

The earlier proof showing duality between c.e. boolean algebras and Π^0_1 classes uses the equivalence of Π^0_1 classes and co-c.e. subsets of the Cantor set. Here we will discuss the Stone duality again but this time using the equivalence of Π^0_1 classes and effectively closed subsets of the Cantor space $\mathbf{C}$, in the meantime also giving a propositional interpretation of Stone duality. With this we will prove a stronger version of Stone duality. Classical Stone duality only gives a natural isomorphism between a boolean algebra and its double dual, and between a boolean space and its double dual. Recall $\mathcal{S}$ is the free boolean algebra on the computable set of free generators P. We will use a canonical homemomorphism between $\mathcal{S}^*$ and $\mathbf{C}$ to define a bijection Φ between quotients of $\mathcal{S}$ and closed subsets of $\mathbf{C}$. This bijection concretely witnesses Stone duality for boolean algebras in the sense that every countable boolean algebra B can be presented as a quotient of $\mathcal{S}$ and $\Phi(B)$ is isomorphic to B^*. Furthermore under Φ, c.e. boolean algebras correspond exactly to effectively closed subsets of $\mathbf{C}$.

Before we define Φ, we find a canonical homeomorphism between $\mathcal{S}^*$ and $\mathbf{C}$. Since $\mathcal{S}^*$ is the set of 2-valued boolean homomorphisms on $\mathcal{S}$ and $\mathcal{S}$ is freely generated by P, we can view the dual space $\mathcal{S}^*$ as the set of 2-valued assignments 2^P. Note $\mathcal{S}^*$ has clopen basis $\{s^* : s \in S(P)\}$. Hence we can define the following map between $\mathcal{S}^*$ and the Cantor space: viewing the Cantor space $\mathbf{C}$ as a set of infinite binary strings, define $\varphi : \mathcal{S}^* \to \mathbf{C}$, $a \mapsto (a(p_i) : i \in \mathbb{N})$ for each 2-valued truth valuation on $\mathcal{S}$.

Proposition 2.16. *φ is a computable homeomorphism between $\mathcal{S}^*$ and $\mathbf{C}$.*

Proof. For disjoint finite subsets U, V of P, let

$$N(U,V) = \left\{a \in 2^P : a(U) = \{1\},\ a(V) = \{0\}\right\},$$

i.e. $N(U,V)$ is the set of all 2-valued truth assignments making U true and V false. Note $N(U,V)$ is clopen, since

$$N(U,V) = \bigcap_{u \in U} u^* \cap \bigcap_{v \in V} (\neg v)^*.$$

For $s \in S(P)$, let W be the set of propositional letters in s. For each 2-valued truth valuation which sends s to 1, there is a 2-valued truth assignment a of W. Since W is finite, there are only finitely many such truth assignments of W. Let

$$U_a = \{p_i \in W : a(p_i) = 1\},$$
$$V_a = \{p_i \in W : a(p_i) = 0\},$$
$$F = \left\{a \in 2^W : a(s) = 1\right\}.$$

Then $s^* = \bigcup_{a \in F} N(U_a, V_a)$. Hence the $N(U,V)$ form a clopen basis of $\mathcal{S}^*$.

Note that φ is bijective, so we need only show that it is computably continuous. $\mathbf{C}$ has topology spanned by $I(\sigma)$ for finite strings σ. For any finite string σ, letting $U = \{p_i : \sigma(i) = 1\}$ and $V = \{p_i : \sigma(i) = 0\}$, we have $\varphi^{-1}(I(\sigma)) = N(U,V)$, which concludes the proof. □

If A is a boolean algebra with domain $S(P)$, a truth assignment on A extends to a truth valuation on $\mathcal{S}$, so A^* is a subspace of $\mathcal{S}^*$. Therefore φ is well-defined on A and is computably continuous. Hence this duality between finite truth assignments and clopen sets extends to c.e. boolean algebras with domain $S(P)$ but a different equivalence.

Definition 2.17. We define Φ and Ψ as follows. For a boolean algebra A with domain $S(P)$, let $\Phi(A)$ be the image of A^* under φ as a subset of $\mathbf{C}$. For a closed subset K of $\mathbf{C}$, let $\Psi(K) = (S(P), \wedge, \vee, \neg, \equiv_K)$, where $u \equiv_K v$ if and only if $u^* \cap K = v^* \cap K$.

We showed in Theorem 2.12 that if K is effectively closed, then $\Psi(K)$ is a c.e. boolean algebra. We will show that if A is c.e., then $\Phi(A)$ is an effectively closed subset of $\mathbf{C}$.

Proposition 2.18. *If A is a c.e. boolean algebra with domain $S(P)$, then $\Phi(A)$ is an effectively closed subset of $\mathbf{C}$. Further $\Phi(A)$ is computably homeomorphic to A^*.*

Proof. We will reuse the map φ and notation $N(U, V)$ from the proof of Proposition 2.16. We will show $K = \Phi(A)$ is effectively closed.

Since we can computably enumerate the elements equivalent to 1 in A, we can also computably enumerate the finite truth assignments sending at least one such element to 0. Each finite truth assignment corresponds to an $N(U, V)$ which is disjoint from K. The union of all these $N(U, V)$ is the complement of K. Since $N(U, V)$ is clopen, the complement of K is open, so K is effectively closed, as desired.

That $\Phi(A)$ is homeomorphic to A^* follows from Proposition 2.16. □

Using the technique of the previous proposition, Stone duality in fact gives a bijection between all countable boolean algebras and closed subsets of the Cantor space. Under that bijection, c.e. boolean algebras correspond exactly to Π^0_1 classes.

Theorem 2.19. *Φ is a canonical bijection between c.e. boolean algebras and Π^0_1 classes.*

Proof. Let A be a c.e. boolean algebra. Then $\Psi(\Phi(A))$ is $(S(P), \wedge, \vee, \neg, \equiv_{\Phi(A)})$, where $p \equiv_{\Phi(A)} q$ if and only if $p^* \cap \Phi(A) = q^* \cap \Phi(A)$. This is equivalent to the condition that for every ultrafilter f of A, f is in p^* if and only if f is in q^*. By definition, this is equivalent to saying that $f(p) = 1$ if and only if $f(q) = 1$. However for every pair of distinct elements, there is an ultrafilter which separates them, so this means $p \equiv_A q$. Hence $\Psi(\Phi(A)) = A$.

Let K be an effectively closed subset of Cantor space. Then $\Phi(\Psi(K))$ is the set of ultrafilters of $(S(P), \wedge, \vee, \neg, \equiv_K)$. This is exactly the set of ultrafilters f on $\mathcal{S}$ such that if $u \equiv_K 1$, then $f(u) = 1$, or equivalently, if $u^* \supseteq K$, then $f(u) = 1$. By definition of u^*, this is the set of ultrafilters f on $\mathcal{S}$ such that if $u^* \supseteq K$, then $f \in u^*$. By Proposition 2.2, since the Cantor set can be thought of as $\mathcal{S}^*$, this is the set of ultrafilters f on $\mathcal{S}$ such that if U is a clopen subset of Cantor space and $U \supseteq K$, then $f \in U$. Because the Cantor space is Hausdorff and K is compact, for every $f \notin K$, there is a clopen set of the Cantor space which contains K but not f, so finally we can conclude that $\Phi(\Psi(K)) = K$.

Since we have shown that $\Psi \circ \Phi$ and $\Phi \circ \Psi$ are identity maps, they are bijections between c.e. boolean algebras and Π^0_1 classes. □

Much of the above proof works for boolean algebras and boolean spaces, except the need for some kind of "universal" boolean space to serve the role of the Cantor space when showing that $\Psi(\Phi(K)) = K$. It would be interesting to consider whether this obstacle can be overcome for general boolean algebras and boolean spaces.

2.6 Examples

To conclude this chapter, we discuss a couple concrete examples of c.e. boolean algebras and Π^0_1 classes. Of course one obvious example, as a result of the duality, is the boolean algebra of all clopen subsets of an effectively closed subset of the Cantor set. Even without Stone duality, we can also see that the Lindenbaum algebra of number theory based on the Peano axioms is c.e. but not computable, nor are its proper (consistent) c.e. homomorphic images. If it were, the preimage of 1 would be a maximal computable filter and hence a computable set separating provable and refutable statements, contradicting the second incompleteness theorem.

A more interesting consequence of the duality is the existence of c.e. boolean algebras with no computable ultrafilters, which follows from the existence of nonempty Π^0_1 subsets of the Cantor set with no computable points.

Proposition 2.20. *There exist c.e. boolean algebras with no computable ultrafilters.*

Proof. Let K be a nonempty Π^0_1 subset of the Cantor set with no computable points, and let $\mathcal{B}$ be its Stone dual, a c.e. boolean algebra. We will show that a computable ultrafilter F in $\mathcal{B}$ corresponds to a computable point in K, and since K has no computable points, $\mathcal{B}$ has no computable ultrafilters.

Note that a Π^0_1 set with no computable points corresponds bijectively with a Π^0_1 subset of the Cantor set with no computable points, and a point in the Cantor set corresponds bijectively to an infinite path through the infinite binary tree. For a finite string σ in the tree, the basic clopen set $I(\sigma)$ is the set of infinite paths passing through the node corresponding to σ. Therefore an infinite path

corresponds to a collection of basic clopen sets, which generates an ultrafilter. Though this is not the direction we need, it illustrates how we will think in the next part.

To show that F corresponds to an infinite path through the tree, it suffices to show that if $I(\sigma) \cup I(\tau)$ is in F, where $|\sigma| = |\tau|$ and $\sigma \neq \tau$, then either $I(\sigma)$ or $I(\tau)$ is in F but not both. First $I(\sigma) \cap I(\tau) = \varnothing$ which is not in F, so it cannot be that both are in F. Suppose $I(\sigma)$ is not in F. Then $\neg I(\sigma)$ is in F, and so is $\neg I(\sigma) \cap (I(\sigma) \cup I(\tau)) = \neg I(\sigma) \cup I(\tau)$. Then also in F is

$$(\neg I(\sigma) \cup I(\tau)) \cap (I(\sigma) \cup I(\tau)) = I(\tau),$$

as desired.

Finally it remains to show that if F is computable, so is the infinite path. This is indeed the case, as the path can be found by starting at root node and traversing to whichever child node is in F. Exactly one of the child nodes will be in F, since the current node is in F and F is an ultrafilter. Since membership in F is computable, so is determining the path to arbitrary length, which corresponds to computing the point to arbitrary precision. □

We will discuss computably universal c.e. boolean algebras in greater detail in Chapter 3. However because it fits the nature of this chapter better, we will mention the following result, which links computably universal c.e. boolean algebras to Medvedev-complete Π^0_1 classes.

Definition 2.21. For subsets P, Q of Cantor space, P is *Medvedev-reducible* to Q, denoted $P \leq_M Q$, if there is a partial computable Φ which is defined for all $X \in Q$ and maps Q into P. Q is *Medvedev-complete* if for every nonempty Π^0_1 class P, $P \leq_M Q$.

Proposition 2.22. *If A is a computably universal c.e. boolean algebra, then A^* is a Medvedev-complete Π^0_1 class.*

Proof. Suppose K is a nonempty Π^0_1 class, and let B be its corresponding c.e. boolean algebra as described by Theorem 2.19. Since K is nonempty, B is non-degenerate. A is computably universal, so there is a computable injective homomorphism f from B into A. As shown in Theorem 2.15, f^* is a surjective computably continuous map from A^* onto $B^* = K$. Thus A^* is a Medvedev-complete Π^0_1 class. □

Classes of separating sets for pairs of c.e. sets can be used to generate concrete examples of Π^0_1 classes. These results suggest that using Stone duality, c.e. boolean algebras may be a useful alternative for this, including Medvedev-complete Π^0_1 classes. We will not pursue this possibility further here.

2.7 Future work

A natural future direction is to examine the existing (and expansive) literature on Π^0_1 classes, for example by Cenzer and Remmel [2] or Jockusch and Soare [6] [5]. As we did in Proposition 2.20, the duals of these results could yield insights into c.e. boolean algebras which are not obvious from the boolean algebra perspective.

CHAPTER 3

COMPUTABLY UNIVERSAL HOMOGENEITY

We aim to extend the theory of the class of c.e. subsets of $\mathbb{N}$ to the class of c.e. boolean algebras, as well as briefly prove computable versions of classical boolean algebra results, before getting to the main theorem. Because we are keeping an eye towards a computably universal-homogeneous boolean algebra for the class of c.e. boolean algebras, we will think of c.e. boolean algebras as having domain a (computable) free boolean algebra on countably many generators.

Most theorems we give assert that algorithms exist for computing indices of various objects from indices of other objects. We will suppress reference to the algorithms for simplicity of exposition.

3.1 Computably universal-homogeneous c.e. subsets of $\mathbb{N}$

Let $(W_e : e \in \mathbb{N})$ be the standard universal enumeration of c.e. sets of integers. Post's 1945 paper defines a creative set C as a c.e. set for which there exists a partial computable function f such that if W_e is disjoint from C, then $f(e) \notin C \cup W_e$. Post noticed that if we assign Gödel numbers to statements of standard undecidable theories such as arithmetic or set theory, the set of all Gödel numbers of theorems is a creative set. His original example of a creative set was defined as follows. Let $j : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a bijective computable pairing function. Then $K = \{j(x, e) : x \in W_e\}$ is creative. For fixed y, $\{j(x, e) : x \in W_e\}$ is a copy of W_y. So K is an "effectively disjoint" union of one copy of each W_y. In 1952 Myhill showed the following:

Theorem 3.1 (Myhill's theorem)**.** *Any two creative sets differ by a computable permutation. For creative set C and any c.e. set B there is an injective computable total function $f : \mathbb{N} \to \mathbb{N}$ such that $x \in B \iff f(x) \in C$.*

This asserts that creative sets are determined up to a computable permutation. The key to his proof was using Kleene's recursion theorem to prove:

Lemma 3.2. *Suppose $B \leq_1 C$ and we have an injective finite map g from a subset D of $\mathbb{N}$ to $\mathbb{N}$ such that $x \in B \iff g(x) \in C$. For any integer $y \notin D$, we can compute a $z \notin g(D)$ such that $y \in B$ if and only if $z \in C$.*

A "back and forth" argument and a "forth" argument yield Myhill's theorem from this lemma.

3.2 Analog for c.e. boolean algebras

To give some context to the following results, we present two definitions and their computable and c.e. counterparts.

Definition 3.3. A boolean algebra X is *universal* for a class $\mathcal{K}$ of boolean algebras if X is in $\mathcal{K}$, and for any $B \in \mathcal{K}$, there is an injective homomorphism from B into X. X is *universal-homogeneous* for $\mathcal{K}$ if $X \in \mathcal{K}$, and given

1. a boolean algebra B,
2. a finite subset E of B which generates a subalgebra B' of B,
3. an injective homomorphism $g : B' \to X$, and
4. any $b \in B$,

and letting B'' be the subalgebra of B generated by B' and b, then there is an injective homomorphism $g' : B'' \to X$ which extends g.

Note that being universal-homogeneous is stronger than being universal: if X is universal-homogeneous for $\mathcal{K}$, then X is universal for $\mathcal{K}$. We modify these descriptions slightly to suit our context.

Definition 3.4. A boolean algebra X is *computably universal* for a class $\mathcal{K}$ of c.e. boolean algebras if X is in $\mathcal{K}$, and given (an index of) any $B \in \mathcal{K}$, we can compute (an index of) a computable injective homomorphism from B into X. X is *computably universal-homogeneous* for $\mathcal{K}$ if $X \in \mathcal{K}$, and given

1. (an index of) a c.e. boolean algebra B,
2. a finite subset E of B which generates a subalgebra B' of B,
3. (an index of) an injective homomorphism $g : B' \to X$, and
4. any $b \in B$,

and letting B'' be the subalgebra of B generated by B' and b, we can compute (an index of) an injective homomorphism $g' : B'' \to X$ extending g.

This notion of computably universal-homogeneous can generalize to any algebraic structure. With this language, Lemma 3.2 can be restated as, creative sets are computably universal-homogeneous for the class of c.e. subsets of $\mathbb{N}$, and Theorem 3.1 as saying that they are universal for the class. We will follow Myhill's general strategy of using universal-homogeneity to prove universal. However our proof for computably universal-homogeneous is not analogous to Myhill's proof for $\mathbb{N}$. The increased structure of boolean algebras requires additional conditions and finesse.

3.3 Classical and computable boolean algebra

Before we dive into c.e. boolean algebras, we look at the classical and computable analogs. They are surprisingly similar to each other, whereas the computably enumerable case takes a different form. This is an indication of how strong a condition computability is, and how much more interesting things get when we have only computable enumerability.

For both the class of countable boolean algebras and the class of computable boolean algebras, we use the same boolean algebra and show it to be universal-homogeneous and computably universal-homogeneous for each class respectively. Recall $\mathcal{S}$ is a computable free boolean algebra on countably many generators. In classical boolean algebra, we have the following lemma and result.

Lemma 3.5. *Suppose B is a computable countably infinite free boolean algebra (say propositional logic). Let B' be a finite subalgebra of B. Suppose $g : B' \to \mathcal{S}$ is an injective homomorphism and $x \in B$. Then there exists a $y \in \mathcal{S}$ such that if B'' is the subalgebra of B generated by B' and x, then g can be extended to an injective homomorphism $g' : B'' \to \mathcal{S}$ such that $g'(x) = y$.*

Theorem 3.6. *Any two countable atomless boolean algebras are isomorphic. Any countable boolean algebra is monomorphic to a subalgebra of any countable atomless boolean algebra.*

To prove these, we will need a few definitions and basic results. Much of the structure of these proofs is borrowed from Givant and Halmos [4], and the classical proofs can be found there. Though the computable proofs are generally similar to the classical ones, we produce them for completeness. We will also

prove Lemma 3.5 because it demonstrates the proof structure we use for the class of computable boolean algebras and the class of c.e. boolean algebras.

For ease of notation, let

$$b^k = \begin{cases} b & \text{if } k = 1 \\ \neg b & \text{if } k = 0 \end{cases}.$$

Proposition 3.7. *Let $\mathcal{B}$ be the subalgebra generated by a finite subset E of a larger boolean algebra. The atoms of $\mathcal{B}$ are the nonzero elements of the form $p_a = \bigwedge_{e \in E} e^{a(e)}$ for two-valued functions a on E. Every element of $\mathcal{B}$ can be written as the join of a unique subset of these atoms.*

Definition 3.8. A family $\{\mathcal{B}_i\}$ of subalgebras is *directed* if for any $\mathcal{B}_i$ and $\mathcal{B}_j$, there is a $\mathcal{B}_k$ in the family such that $\mathcal{B}_i$ and $\mathcal{B}_j$ are both subalgebras of $\mathcal{B}_k$. A family $\{f_i\}$ of $\mathcal{A}$-valued boolean homomorphisms is *directed* if for any f_i and f_j, there is a f_k in the family such that f_k extends both f_i and f_j.

Proposition 3.9. *The union of a nonempty directed family of boolean algebras is a boolean algebra.*

Proposition 3.10. *A directed family of $\mathcal{A}$-valued homomorphisms always has a common extension to an $\mathcal{A}$-valued homomorphism. If the homomorphisms in the family are injective, then so is the common extension.*

The following two results capture precisely what is necessary to computably extend a homomorphism.

Lemma 3.11 (Homomorphism Extension Criterion)**.** *A mapping g from generating set E of boolean algebra B into a boolean algebra A can be (computably) extended to a homomorphism from B into A exactly if for every finite subset E' of E and two-valued*

function a on E',

$$\bigwedge_{e\in E'} e^{a(e)} = 0 \quad \textit{implies} \quad \bigwedge_{e\in E'} g(e)^{a(e)} = 0.$$

Proof. First suppose g satisfies the criterion and E is finite. Let

$$p_a = \bigwedge_{e\in E} e^{a(e)}, \qquad q_a = \bigwedge_{e\in E} g(e)^{a(e)}.$$

To express joins of these, if X is a subset of 2^E, let

$$p_X = \bigvee_{a\in X} p_a, \qquad q_X = \bigvee_{a\in X} q_a.$$

Then B is finite, its atoms are the nonzero elements of $\{p_a : a \in 2^E\}$, and an element of B can be expressed as p_X for an $X \subseteq 2^E$. Analogous remarks apply to the subalgebra of A generated by $g(E)$, with $\{q_a : a \in 2^E\}$ and q_X. Let $f(p_X) = q_X$ for $X \subseteq 2^E$.

Let $K = \{a \in 2^E : p_a \neq 0\}$ and suppose $p_X = p_Y$. By Proposition 3.7, each $r \in B$ is the join of a unique subset of $\{a \in K : p_a\}$, so $X \cap K = Y \cap K$. For $a \in X - K$ and $b \in Y - K$, $p_a = p_b = 0$, so the criterion ensures $q_a = q_b = 0$, and

$$\bigvee_{a\in X-K} q_a = \bigvee_{a\in Y-K} q_a = 0.$$

Then

$$q_X = \bigvee_{a\in X\cap K} q_a \vee \bigvee_{a\in X-K} q_a = \bigvee_{a\in Y\cap K} q_a \vee \bigvee_{a\in Y-K} q_a = q_Y,$$

which shows f is well-defined.

To show f is a homomorphism, we will show that for subsets X, Y of 2^E, $p_X \vee p_Y = p_{X\cup Y}$ and $p_X \wedge p_Y = p_{X\cap Y}$. The former follows directly from definition. To show the latter:

$$p_X \wedge p_Y = \left(\bigvee_{a\in X} p_a\right) \wedge \left(\bigvee_{b\in Y} p_b\right) = \bigvee_{a\in X, b\in Y} (p_a \wedge p_b) = \bigvee_{a\in X\cap Y} p_a = p_{X\cap Y}.$$

These also hold for q_X and q_Y. Then

$$f(p_X \vee p_Y) = f(p_{X \cup Y}) = q_{X \cup Y} = q_X \vee q_Y.$$

Note

$$p_X \wedge p_{2^E - X} = p_{X \cap (2^E - X)} = p_\varnothing = 0,$$

$$p_X \vee p_{2^E - X} = p_{X \cup (2^E - X)} = p_{2^E} = 1,$$

so $\neg p_X = p_{2^E - X}$. Analogously, $\neg q_X = q_{2^E - X}$. Therefore f is a homomorphism.

To show that f agrees with g on E, we will use one more lemma. If E' is an arbitrary subset of E and b is a two-valued function on E, then

$$p_b = \bigvee \left\{ p_a : a \in 2^E,\ a \text{ extends } b \right\}.$$

To see this, let L be the set of two-valued functions on E which extend b. If $a \in L$, then $p_b \geq p_a$ because $\left\{ e^{b(e)} : e \in E \right\} \subseteq \left\{ e^{a(e)} : e \in E \right\}$. If $a \notin L$, then there is some $e \in E$ such that $a(e) \neq b(e)$, so $\left\{ e^{a(e)}, e^{b(e)} \right\} = \{ e, \neg e \}$ and $p_a \wedge p_b \leq e \wedge \neg e = 0$. Then

$$p_b = p_b \wedge 1 = p_b \wedge \bigvee_{a \in 2^E} p_a = \bigvee_{a \in K} (p_b \wedge p_a) = \bigvee_{a \in L} (p_b \wedge p_a) \vee \bigvee_{a \notin L} (p_b \wedge p_a) = \bigvee_{a \in L} p_a,$$

as desired. Therefore for $e \in E$, we have $e = \bigvee \left\{ p_a : a \in 2^E, a(e) = 1 \right\}$ and $g(e) = \bigvee \left\{ q_a : a \in 2^E, a(e) = 1 \right\}$. Then

$$f(e) = \bigvee \left\{ f(p_a) : a \in 2^E, a(e) = 1 \right\} = \bigvee \left\{ q_a : a \in 2^E, a(e) = 1 \right\} = g(e),$$

so f is a homomorphism extending g when E is finite.

Now suppose E is an arbitrary generating set. For each finite subset E' of E, let $B_{E'}$ be the subalgebra generated by E' and $g_{E'}$ be the restriction of g to E'. Because g satisfies the criterion, so does $g_{E'}$, so each $g_{E'}$ can be extended to a

homomorphism $f_{E'}$ defined on $B_{E'}$. The $f_{E'}$ form a directed family of A-valued homomorphisms, so by Proposition 3.10 there's a common extension f of all $f_{E'}$ with domain $\bigcup_{E'} B_{E'} = B$. This f is a homomorphism extending every $g_{E'}$ so it extends g (finally).

To show the converse, suppose a mapping $g : E \to A$ can be extended to a homomorphism $f : B \to A$, E' is a finite subset of E, and a is a two-valued function on E'. If $\bigwedge_{e \in E'} e^{a(e)} = 0$, then

$$0 = f(0) = f\left(\bigwedge_{e \in E'} e^{a(e)}\right) = \bigwedge_{e \in E'} f(e)^{a(e)} = \bigwedge_{e \in E'} g(e)^{a(e)},$$

concluding the proof. □

Lemma 3.12 (Monomorphism Extension Criterion)**.** *A mapping g from generating set E of boolean algebra B into a boolean algebra A can be (computably) extended to an injective homomorphism from B into A exactly if for every finite subset E' of E and two-valued function a on E',*

$$\bigwedge_{e \in E'} e^{a(e)} = 0 \quad \textit{if and only if} \quad \bigwedge_{e \in E'} g(e)^{a(e)} = 0.$$

Proof. The proof for this is very similar to that of Lemma 3.11, so we will merely point out where the proof differs, using much of the same notation such as p_a, q_a, p_X, q_X, and K. As before, first assume E is finite. Previously, for subsets X, Y of 2^E, we had $p_X = p_Y$ if and only if $X \cap K = Y \cap K$, and if $X \cap K = Y \cap K$, then $q_X = q_Y$. With the stronger criterion, we can now say that K is also the set of two-valued functions a on E such that $q_a \neq 0$. Therefore $q_X = q_Y$ if and only if $X \cap K = Y \cap K$, so $p_X = p_Y$ if and only if $q_X = q_Y$. Letting $f(p_X) = q_X$, we therefore have that f is an injective homomorphism extending g if E is finite. The proof for arbitrary E is the same but using the injective case of Proposition 3.10.

To prove the converse, note that since f is a injection, $p_a = 0$ if and only if $f(p_a) = 0$. Otherwise this is identical to before. □

To restate, to computably extend an injective homomorphism, it suffices to demonstrate that the existing map sends a "potential atom," a conjunction of generators or their negations, to 0 if and only if its preimage is 0.

Lemma 3.13. *Let B be a computable boolean algebra with elements $\{b_i : i \in \mathbb{N}\}$, and let B_n be the (finite) subalgebra generated by $\{b_0, \ldots, b_{n-1}\}$. Suppose $g : B_n \to \mathcal{S}$ is a computable injective homomorphism. Then g can be computably extended to a computable injective homomorphism $g' : B_{n+1} \to \mathcal{S}$.*

Proof. For every two-valued function a on $\{0, \ldots, n-1\}$,

$$q = \bigwedge_{i=0}^{n-1} b_i^{a(i)}$$

is either 0 or an atom of B_n. Since g is injective, $q = 0$ if and only if $g(q) = 0$ so the criterion of Lemma 3.12 can be formulated as follows: for every atom q in B_n, $q \wedge b_n \equiv_B 0$ if and only if $g(q) \wedge x \equiv_S 0$, and $q \wedge \neg b_n \equiv_B 0$ if and only if $g(q) \wedge \neg x \equiv_S 0$, where x is the desired image of b_n.

Since B_n is finite, $g(B_n)$ involves only finitely many propositional letters in $\mathcal{S}$, so let r be a propositional letter in P not mentioned in $g(B_n)$. We can find r computably because B_n is finite and g is computable. For each atom q in B_n, define x_q as follows: if $q \wedge b_n$ is equivalent to 0 or q, let $x_q = g(q \wedge b_n)$. Else let $x_q = g(q) \wedge r$. Since r is not mentioned in $g(B_n)$ and $\mathcal{S}$ is free, $g(q) \wedge r$ is strictly below $g(q)$ unless $g(q) \equiv_B 0$. Then $q \wedge b_n \equiv_B 0$ if and only if $g(q) \wedge x_q \equiv_S 0$, and $q \wedge \neg b_n \equiv_B 0$ if and only if $g(q) \wedge \neg x_q \equiv_S 0$. The latter biconditional holds

because $q \wedge b_n \equiv_B q$ if and only if $q \wedge \neg b_n \equiv_B 0$. Let

$$x = \bigvee_{q \text{ atom}} x_q.$$

For every atom q of B_n, $g(q) \wedge x \equiv_S g(q) \wedge x_q$, and $g(q) \wedge \neg x \equiv_S g(q) \wedge \neg x_q$. So let h be the map on $B_n \cup \{b_n\}$ such that $h(b) = g(b)$ for $b \in B_n$ and $h(b_n) = x$. We have shown that h satisfies the criterion for lemma 3.12 and $B_n \cup \{b_n\}$ generates B_{n+1}, so h extends to an injective homomorphism of B_{n+1} into $\mathcal{S}$. □

Then we have the following two theorems.

Theorem 3.14. *$\mathcal{S}$ is a computably universal-homogeneous boolean algebra for the class of computable boolean algebras.*

Theorem 3.15. *Any nondegenerate computable boolean algebra can be computably embedded in a computable atomless boolean algebra.*

The computable version of Theorem 3.6 has a nearly identical proof.

Theorem 3.16. *Any two nondegenerate computable atomless boolean algebras are computably isomorphic.*

Proof. We will use a standard "back-and-forth" argument.

Let A and B be two computable atomless boolean algebras, with enumerations $a_1, a_3, a_5, \ldots$ and $b_2, b_4, b_6, \ldots$ respectively. We will computably define a_n for even n and b_n for odd n so that the map taking a_n to b_n will satisfy the criterion in Lemma 3.12. For finite n let A_n and B_n be the subalgebras of A and B generated by $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ respectively. Since we are constructing a_n and b_n as we go, so are A_n and B_n. After we do this, we have a computable

isomorphism between the subalgebra of A generated by $\{a_n : n \geq 1\}$ to the subalgebra of B generated by $\{b_n : n \geq 1\}$. Since the former includes every element of A and the latter includes every element of B, this gives us a computable isomorphism between A and B as desired.

Let g_0 be the isomorphism from the two-element subalgebra of B to the two-element subalgebra of A. Note that g_0 is computable and finite, and equality is computable in both, so its inverse is a computable injective homomorphism from the two-element subalgebra of A into B. By Lemma 3.13, g_0^{-1} can be computably extended to a computable injective homomorphism g_1 from A_1 into B, and $g_1(A_1) = B_1$.

We have the computable isomorphism g_1 from A_1 into B_1. Since g_1 is computable, B_1 is finite, and equality is computable, g_1^{-1} is a computable injection from B_1 into A. As before, by Lemma 3.13, g_1^{-1} can be computably extended to a computable injection g_2 from B_2 into A. Letting $a_2 = g_2(b_2)$, we then have $g_2(B_2) = A_2$.

Now suppose g_n is a computable injective homomorphism. If n is odd, then g_n is from A_n into B. Since g_n is computable and its domain is finite, g_n^{-1} is a computable injective homomorphism from $g_n(A_n) = B_n$ into A. As before g_n^{-1} can be computably extended to a computable injection g_{n+1} from B_{n+1} into A. Let $b_{n+1} = g_{n+1}(a_{n+1})$, so $g_{n+1}(B_{n+1}) = A_{n+1}$, completing this step.

An analogous construction holds for the case if n is even and g_n is from B_n into A, which finishes the proof. □

3.4 Computably enumerable boolean algebra

We finally arrive at the computably enumerable case. As hinted before, this will not be as simple as the classical and computable cases. The following construction will appear strange but the proofs and subsequent discussion will hopefully motivate it in hindsight.

Let K be the boolean algebra generated by Turing machines. If a Turing machine p halts and outputs 0 or 1, add $p \equiv_K 0$ or $p \equiv_K 1$ respectively to the equivalence relation. Other Turing machines are free generators. The equivalence relation generated by this is c.e., so K is a nondegenerate c.e. boolean algebra.

Lemma 3.17. *Let K be as defined above. Let B be an arbitrary c.e. boolean algebra with elements $\{b_i : i \in \mathbb{N}\}$, and let B_n be the (finite) subalgebra generated by $\{b_0, \ldots, b_{n-1}\}$. If $g : B_n \to K$ is a computable injective homomorphism, then g can be computably extended to a computable injective homomorphism $g' : B_{n+1} \to K$.*

Proof. For every two-valued function a on $\{0, \ldots, n-1\}$,

$$q = \bigwedge_{i=0}^{n-1} b_i^{a(i)}$$

is either 0 or an atom of B_n. We will call these "potential atoms" of B_n. Since g is injective, the criterion of Lemma 3.12 can be formulated as follows: for every potential atom q in B_n, $q \wedge b_n \equiv_B 0$ if and only if $g(q) \wedge y \equiv_K 0$, and $q \wedge \neg b_n \equiv_B 0$ if and only if $g(q) \wedge \neg y \equiv_K 0$, where y is the desired image of b_n.

For each potential atom q in B_n, let x_q be the Turing machine which runs the algorithms to check both $q \wedge b_n \equiv_B 0$ and $q \wedge \neg b_n \equiv_B 0$, alternating steps from each. If the algorithm checking $q \wedge b_n \equiv_B 0$ halts first and answers in the

affirmative, then x_q halts and outputs 0. If the algorithm checking $q \wedge \neg b_n \equiv_B 0$ halts first and answers in the affirmative, then x_q halts and outputs 1. Thus

$$x_q \equiv_K 0 \qquad \text{if and only if} \qquad q \wedge b_n \equiv_B 0$$
$$\neg x_q \equiv_K 0 \qquad \text{if and only if} \qquad q \wedge \neg b_n \equiv_B 0.$$

If x_q is one of the generators mentioned in $g(q)$, then let x_q be an equivalent Turing machine not mentioned in $g(q)$.

Since x_q is not mentioned in $g(B_n)$ and Turing machines which do not halt are free generators, $g(q) \wedge x_q \equiv_K 0$ if and only if $g(q) \equiv_B 0$ or $x_q \equiv_K 0$, which occurs if and only if $q \wedge b_n \equiv_B 0$. Furthermore $g(q) \wedge x_q$ is strictly below $g(q)$ unless $g(q) \equiv_B 0$ or $q \wedge \neg b_n \equiv_B 0$. Then $q \wedge \neg b_n \equiv_B 0$ if and only if $g(q) \wedge \neg x_q \equiv_K 0$. Let

$$x = \bigvee_{q \text{ potential atom}} (g(q) \wedge x_q).$$

For every potential atom q of B_n, $g(q) \wedge x \equiv_K g(q) \wedge x_q$, and $g(q) \wedge \neg x \equiv_K g(q) \wedge \neg x_q$. So let h be the map on $B_n \cup \{b_n\}$ such that $h(b) = g(b)$ for $b \in B_n$ and $h(b_n) = x$. We have shown that h satisfies the criterion for lemma 3.12 and $B_n \cup \{b_n\}$ generates B_{n+1}, so h extends to an injective homomorphism of B_{n+1} into K. □

The construction of K is contrived but highlights useful properties for showing that a c.e. boolean algebra is computably universal-homogeneous. We need an algorithm to computably generate an element of the boolean algebra which is free from a given element if the element is neither 0 nor 1 without knowing in advance if the element is. And we need a way to run Turing machines. The following definition encapsulates these requirements.

Definition 3.18. A boolean algebra $(A, \wedge, \vee, \neg, \equiv)$ is a *Rosser-Turing boolean algebra* if there are computable maps $r, t : \mathbb{N} \to A$ such that:

1. if the eth c.e. subset, W_e, of A is consistent, i.e. the equivalence relation $\equiv'$ gotten by extending $\equiv$ with $p \equiv' 1$ for $p \in W_e$ is nondegenerate, then both $W_e \cup \{r(e)\}$ and $W_e \cup \{\neg r(e)\}$ are also consistent;
2. $t(e) \equiv 1_A$ if and only if the eth Turing machine halts and outputs 1, and
3. $t(e) \equiv 0_A$ if and only if the eth Turing machine halts and outputs 0.

We call r and t Rosser and Turing maps respectively of $\mathcal{A}$.

The first requirement refers to the computational content of the second incompleteness theorem. This might be easier to see if we switch to the language of quotients: given a filter, f outputs a nonzero element not in the filter. The second and third requirements say the boolean algebra must be able to "run" Turing machines. In our proofs we use the latter two to check equivalence and the first to generate an independent element with which we construct the image.

Though this definition of a Rosser-Turing boolean algebra does not require the boolean algebra to be c.e., from here on, unless otherwise stated, we are only considering c.e. boolean algebras.

Proposition 3.19. *If B is a Rosser-Turing boolean algebra and h is a computable injective homomorphism from B into A, then A is also a Rosser-Turing boolean algebra.*

Proof. Since B is a Rosser-Turing boolean algebra, it has computable Rosser and Turing maps r and t respectively. Since h is injective, $h(t(e)) \equiv_A 1$ if and only if $t(e) \equiv_B 1$ if and only if the eth Turing machine halts and outputs 1, which shows A can run Turing machines.

Though we have deliberately avoided filters so far, it will be clearer and more concise to prove the other property with them. Let F be a c.e. filter of A. Since h

is computable, the preimage of F under h is a c.e. filter of B, with index e. Then $r(e)$ is a nonzero element of B not in $h^{-1}(F)$, and we have $h(r(e))$ is the desired nonzero element of A not in F. Hence A is a Rosser-Turing boolean algebra. □

An immediate consequence is that any boolean algebra with a Rosser-Turing subalgebra is also Rosser-Turing. Another is that free products, so long as they are nondegenerate, also preserve Rosser-Turing boolean algebras.

Proposition 3.20. *The free product of a Rosser-Turing boolean algebra with any nondegenerate c.e. boolean algebra is also a Rosser-Turing boolean algebra.*

Proof. This follows from the categorical definition of a free product. □

Without further ado, we will show that a Rosser-Turing boolean algebra is computably universal-homogeneous for the class of c.e. boolean algebras, following the same outline as for the countable and computable cases.

Lemma 3.21. *Let A be a Rosser-Turing boolean algebra with Rosser and Turing maps r and t respectively. Let B be an arbitrary c.e. boolean algebra with elements $\{b_i : i \in \mathbb{N}\}$, and let B_n be the (finite) subalgebra generated by $\{b_0, \ldots, b_{n-1}\}$. If $g : B_n \to A$ is a computable injective homomorphism, then g can be computably extended to a computable injective homomorphism $g' : B_{n+1} \to A$.*

Proof. The proof structure is the same as before; in fact, after defining x_q, the proof is identical.

For every two-valued function a on $\{0, \ldots, n-1\}$,

$$q = \bigwedge_{i=0}^{n-1} b_i^{a(i)}$$

is a "potential atom" of B_n. The criterion of Lemma 3.12 can be formulated as follows: for every potential atom q in B_n, $q \wedge b_n \equiv_B 0$ if and only if $g(q) \wedge y \equiv_A 0$, and $q \wedge \neg b_n \equiv_B 0$ if and only if $g(q) \wedge \neg y \equiv_A 0$, where y is the desired image of b_n. The latter biconditional is equivalent to $q \wedge b_n \equiv_B q$ if and only if $g(q) \wedge y \equiv_A g(q)$.

For a Turing machine m, we will use the notation $[m]$ to indicate $t(m)$, even if we do not state m explicitly. For instance, if b, c are elements of B, $[b \equiv_B c]$ means t applied to the Turing machine which tests equality of b and c in B, so it is equivalent to 1_A if the equality holds. Since equality in B is c.e., the sentence determines whether any run of the associated Turing machine halts and answers yes.

For an element b of B_n, we use the recursion theorem to define a Turing machine $M_n(b)$ which tests the following four equalities simultaneously:

$$b \wedge b_n \equiv_B 0 \qquad g(b) \wedge \neg[M_n(b) = 1] \equiv_A 0 \tag{3.1}$$

$$b \wedge b_n \equiv_B b \qquad g(b) \wedge \neg[M_n(b) = 0] \equiv_A 0. \tag{3.2}$$

If either equality in line 3.1 answers yes, then $M_n(b)$ outputs 0; if either equality in line 3.2 answers yes, then $M_n(b)$ outputs 1.

Let $\rho_b = r(g(b) \wedge \neg[M_n(b) = 0] \wedge \neg[M_n(b) = 1])$. For instance, if $g(b) \not\equiv_A 0$, then $g(b) \wedge \rho_b$ nor $g(b) \wedge \neg\rho_b$ is equivalent to 0. Equivalently, if $g(b) \not\equiv_A 0$, then $0 < g(b) \wedge \rho_b < g(b)$ in A.

For each potential atom q, let

$$\begin{aligned} x_q = &([M_n(q) = 0] \Rightarrow 0) \wedge ([M_n(q) = 1] \Rightarrow 1) \\ &\wedge (\neg([M_n(q) = 0] \vee [M_n(q) = 1]) \Rightarrow \rho_q) \\ \equiv_A &\neg[M_n(q) = 0] \wedge ([M_n(q) = 1] \vee \rho_q). \end{aligned}$$

Note that

$$x_q \equiv_A [M_n(q) = 1] \vee (\neg[M_n(q) = 0] \wedge \rho_q).$$

As before, it suffices to show

$$\begin{aligned} q \wedge b_n \equiv_B 0 \qquad & \text{if and only if} \qquad g(q) \wedge x_q \equiv_A 0, \\ q \wedge b_n \equiv_B q \qquad & \text{if and only if} \qquad g(q) \wedge x_q \equiv_A g(q). \end{aligned}$$

Note that if $q \wedge b_n \equiv_B 0$, then $x_q \equiv_A 0$, and if $q \wedge b_n \equiv_B q$, then $x_q \equiv_A 1$, so it remains to show that if $0 < q \wedge b_n < q$, then $0 < g(q) \wedge x_q < g(q)$. Equivalently, $g(q) \wedge x_q$ and $g(q) \wedge \neg x_q$ are nonzero. Since $0 < q \wedge b_n < q$, we "know" that $g(q)$, $\neg[M_n(q) = 0]$, and $\neg[M_n(q) = 1]$ are all nonzero.

First suppose $\neg[M_n(q) = 0]$ and $\neg[M_n(q) = 1]$ are both consistent with $g(q)$. Equivalently $g(q) \wedge \neg[M_n(q) = 0] \wedge \neg[M_n(q) = 1] \not\equiv_A 0$, since $g(q) \not\equiv_A 0$. Then

$$\begin{aligned} g(q) \wedge x_q &\equiv_A g(q) \wedge \neg[M_n(q) = 0] \wedge ([M_n(q) = 1] \vee \rho_q) \\ &\equiv_A (g(q) \wedge \neg[M_n(q) = 0 \wedge [M_n(q) = 1]) \\ &\qquad \vee (g(q) \wedge \neg[M_n(q) = 0] \wedge \rho_q) \\ &\geq_A g(q) \wedge \neg[M_n(q) = 0] \\ &\geq_A g(q) \wedge \neg[M_n(q) = 0] \wedge \neg[M_n(q) = 1] \not\equiv_A 0, \end{aligned}$$

and

$$\begin{aligned} g(q) \wedge \neg x_q &\equiv_A g(q) \wedge ([M_n(q) = 0] \vee (\neg[M_n(q) = 1] \wedge \neg\rho_q) \\ &\equiv_A (g(q) \wedge [M_n(q) = 0]) \vee (g(q) \wedge \neg[M_n(q) = 1] \wedge \neg\rho_q) \\ &\geq_A g(q) \wedge \neg[M_n(q) = 1] \wedge \neg\rho_q \not\equiv_A 0, \end{aligned}$$

as desired.

Now suppose $g(q) \wedge \neg[M_n(q) = 0] \equiv_A 0$. Since $M_n(q)$ checks that as one of the four algorithms it's running simultaneously, if that halts first, then $M_n(q)$ will

output 1. Then $[M_n(q) = 0] \equiv_A 0$, which forces $g(q)$ to be zero, contradicting our assumption. We will also check if that is not the first algorithm to answer yes first. By assumption $q \wedge b_n \not\equiv_B 0$ and $q \wedge b_n \not\equiv_B q$, so the only other algorithm that may answer yes first is the one checking $g(b) \wedge \neg[M_n(q) = 1] \equiv_A 0$. However $[M_n(q) = 0] \Rightarrow \neg[M_n(q) = 1] \equiv_A 1$, or equivalently,

$$[M_n(q) = 0] \wedge [M_n(q) = 1] \equiv_A 0,$$

so if

$$g(q) \wedge \neg[M_n(q) = 0] \equiv_A 0,$$

then

$$g(q) \wedge \neg[M_n(q) = 1] \equiv_A g(q) \not\equiv_A 0,$$

so if $g(q) \wedge \neg[M_n(q) = 0] \equiv_A 0$, then none of the other algorithms $M_n(q)$ checks will halt. Therefore $g(q) \wedge \neg[M_n(q) = 0] \not\equiv_A 0$. An analogous argument also shows that $g(q) \wedge \neg[M_n(q) = 1] \not\equiv_A 0$.

Finally suppose

$$g(q) \wedge \neg[M_n(q) = 0] \wedge \neg[M_n(q) = 1] \equiv_A 0$$

but

$$g(q) \wedge \neg[M_n(q) = 0] \not\equiv_A 0, \qquad g(q) \wedge \neg[M_n(q) = 1] \not\equiv_A 0.$$

Equivalently

$$g(q) \Rightarrow ([M_n(q) = 0] \vee [M_n(q) = 1]) \equiv_A 1$$
$$g(q) \Rightarrow [M_n(q) = 0] \not\equiv_A 1$$
$$g(q) \Rightarrow [M_n(q) = 1] \not\equiv_A 1.$$

From the first we have

$$g(q) \Rightarrow ([M_n(q) = 0] \Leftrightarrow \neg[M_n(q) = 1]) \equiv_A 1.$$

Then we have

$$g(q) \Rightarrow [M_n(q) = 1] \not\equiv_A 1$$
$$g(q) \Rightarrow \neg[M_n(q) = 1] \not\equiv_A 1.$$

Since

$$x_q \wedge [M_n(q) = 1] \equiv_A x_q$$
$$\neg x_q \wedge \neg[M_n(q) = 1] \equiv_A \neg[M_n(q) = 1]$$

we also have

$$g(q) \Rightarrow (x_q \Leftrightarrow [M_n(q) = 1]) \equiv_A 1.$$

and therefore

$$g(q) \Rightarrow x_q \not\equiv_A 1, \qquad g(q) \Rightarrow \neg x_q \not\equiv_A 1.$$

Thus $0 < g(q) \wedge x_q < g(q)$, like we claimed.

Let

$$x = \bigvee_{q \text{ potential atom}} (g(q) \wedge x_q).$$

For every potential atom q of B_n,

$$g(q) \wedge x \equiv_A g(q) \wedge x_q$$
$$g(q) \wedge \neg x \equiv_A g(q) \wedge \neg x_q.$$

So let h be the map on $B_n \cup \{b_n\}$ such that $h(b) = g(b)$ for $b \in B_n$ and $h(b_n) = x$. We have shown that h satisfies the criterion for lemma 3.12 and $B_n \cup \{b_n\}$ generates B_{n+1}, so h extends to an injective homomorphism of B_{n+1} into A. $\square$

As with countable and computable boolean algebras, computable universality and universal-homogeneity follow immediately.

Theorem 3.22. *Let A be a Rosser-Turing boolean algebra. Then A is computably universal-homogeneous for the class of c.e. boolean algebras.*

Theorem 3.23. *Let A be a Rosser-Turing boolean algebra. Every c.e. boolean algebra is computably isomorphic to a subalgebra of A.*

We finish by turning to a more familiar codomain.

Corollary 3.24. *Let A be the Lindenbaum algebra of arithmetic, such as Robinson arithmetic or Peano arithmetic. Then A is computably universal-homogeneous for the class of c.e. boolean algebras.*

Proof. The Rosser condition follows from the second incompleteness theorem. The Turing map exists in any consistent extension of Robinson arithmetic [10]. Then A is a Rosser-Turing boolean algebra and thus computably universal-homogeneous for the class of c.e. boolean algebras. □

To summarize, we have shown that a Rosser-Turing boolean algebra is computably universal-homogeneous for the class of c.e. boolean algebras, and consequently there is only one Rosser-Turing boolean algebra up to computable isomorphism. A grand-sounding corollary is that the Lindenbaum algebra of set theory is computably isomorphic to the Lindenbaum algebra of Robinson arithmetic. While this is itself a neat result, it also makes sense that the Lindenbaum algebra of a c.e. but not computable theory is as complicated as a c.e. boolean algebra can get. This raises the more interesting question of how to characterize c.e. boolean algebras up to computable isomorphism.

3.5 Relation to other work

By imposing more structure, Montagna and Sorbi [7] found computably universal members for a few c.e. structures. They show that every e.i. prelattice is computably universal for the class of computable preorders and that every e.i. boolean prealgebra is computably universal for the class of c.e. preorders and the class of c.e. distributive prelattices. In addition they note that Pour-El and Kripke's work can be generalized to show that every e.i. boolean algebra is computably universal for the class of c.e. boolean algebras. However they do not investigate computably universal-homogeneity.

Because Pour-El and Kripke's result [9] is strikingly similar, also producing a computably universal-homogeneous boolean algebra for the class of c.e. boolean algebras, we will discuss it in more depth.

Definition 3.25. A boolean algebra B is *effectively inseparable*, abbreviated e.i., if there is a computable function φ such that for any e, f, if the eth c.e. set W_e contains all elements of B equivalent to 0, the fth c.e. set W_f contains all elements of B equivalent to 1, and $W_e \cap W_f = \varnothing$, then $\varphi(e, f)$ is an element which is in neither W_e nor W_f.

Because both c.e. Rosser-Turing and e.i. boolean algebras are computably universal-homogeneous for the class of c.e. boolean algebras, using a back-and-forth argument there is a computable isomorphism between any example of each. This can be used to show that if A is a c.e. boolean algebra, then A is e.i. if and only if it is Rosser-Turing. However we will present an alternate proof of the forward direction, which adapts the proof that Pour-El and Kripke uses.

Proposition 3.26. *Suppose A is a c.e. boolean algebra. If A is an e.i. boolean algebra,*

then A is also a Rosser-Turing boolean algebra.

Proof. Note that A satisfies the Rosser condition. Suppose the nth c.e. subset, W_n, is consistent, generating a nondegenerate equivalence relation $\equiv'$. Let $W_e = \{a \in A : a \equiv' 0\}$ and $W_f = \{a \in A : a \equiv' 1\}$. Since W_n is consistent, W_e and W_f are disjoint, so $\varphi(e, f)$ is neither 0 nor 1 under $\equiv'$, and we have our desired element.

To show A also satisfies the Turing condition, we will adapt Pour-El and Kripke's argument. Let m be the Gödel number of a Turing machine, and $[0]$ and $[1]$ denote the elements of A equivalent to 0 and 1 respectively. Via the recursion theorem, let $x = \varphi(e, f)$, where we define W_e and W_f as follows. Let W_e be $[0] \cup \{x\}$ if m halts and outputs 1; otherwise $W_e = [0]$. Let W_f be $[1] \cup \{x\}$ if m halts and outputs 0; otherwise $W_f = [1]$. Note that W_e and W_f are c.e. If m halts and outputs 0, then $W_e = [0]$ and $W_f = [1] \cup \{x\}$. If W_e and W_f were disjoint, then x would be in neither; since x is in W_f, it must be the case that W_e and W_f are not disjoint, which forces $x \equiv_A 0$. Similarly if m halts and outputs 1, then $x \equiv_A 1$. Finally if m does not halt and output 0 or 1, then $W_e = [0]$ and $W_f = [1]$ which are disjoint, so x is equivalent to neither 0 nor 1. Thus the map which sends m to x is a Turing map. □

3.6 Future work

Regarding our characterization of the computably universal-homogeneous c.e. boolean algebras, we wonder if our descriptor could be made more elegant. In both proofs, we use both functions in Definition 3.18. Certainly any Rosser-

Turing boolean algebra is a computably universal-homogeneous c.e. boolean algebra. However we were unable to answer the question, are both properties necessary? In particular is there some way to weaken the requirement of a Turing map, or even get rid of it altogether?

The definition of a c.e. boolean algebra generalizes readily to other structures. We can define a c.e. Birkhoff structure to be one where the domain is $\mathbb{N}$, the functions are computable, and the equivalence relation is computably enumerable. In this context, c.e. structures occur throughout algebra, as presentations are most often given as computable (or c.e.) lists of equations from which further equivalences can be deduced. For example if a countable group's word problem is undecidable, its equivalence relation is c.e. and not computable. Birkhoff's theorem is a tidy result about varieties but is there an analogue for c.e. Birkhoff structures? What classes of structures have a computably universal-homogeneous member? We have answers for vector spaces and boolean algebras but the proof for boolean algebras fails even for distributive lattices.

On the flip side, equality in $\mathbb{R}$ is co-c.e., so it would be equally interesting to consider co-c.e. structures, where once again the domain is $\mathbb{N}$ and the functions are computable, but the equivalence relation is co-c.e.

BIBLIOGRAPHY

[1] Douglas Cenzer. Π^0_1 classes in computability theory. In E.R. Griffor, editor, *Handbook of Computability Theory*, volume 140 of *Studies in Logic and the Foundations of Mathematics*, chapter 2, pages 37–85. Elsevier Science B.V., 1st edition, 1999.

[2] Douglas Cenzer and Jeffrey Remmel. Effectively closed sets, Π^0_1 classes. draft on webpage at `https://people.clas.ufl.edu/cenzer/files/book5.pdf`, September 2014.

[3] Baruch Fischhoff. Hindsight $\neq$ foresight: the effect of outcome knowledge on judgment under uncertainty. *Journal of Experimental Psychology: Human Perception and Performance*, 1:288–299, 1975.

[4] Steven Givant and Paul Halmos. *Introduction to boolean algebras*. Springer Science+Business Media, LLC, 2009.

[5] Carl Jockusch and Robert Soare. Degrees of members of Π^0_1 classes. *Pacific Journal of Mathematics*, 40(3):605–616, 1972.

[6] Carl Jockusch and Robert Soare. Π^0_1 classes and degrees of theories. *Transactions of American Mathematical Society*, 173:33–56, 1972.

[7] Franco Montagna and Andrea Sorbi. Universal recursion theoretic properties of r.e. preordered structures. *Journal of Symbolic Logic*, 50(2):397–406, 1985.

[8] John Myhill. Creative sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 1:97–108, 1955.

[9] Marian Pour-El and Saul Kripke. Deduction-preserving "recursive isomorphism" between theories. *Fundamenta Mathematicae*, 61:141–163, 1967.

[10] Wolfgang Rautenberg. *A concise introduction to mathematical logic (3rd ed)*. Springer Science+Business Media, LLC, 2010.